
Appendix D: Engineering the CCR system

For easier reading throughout this chapter, the combination of the following pieces is referred to as the CCR system.

- a Meridian 1 system with its appropriate software
- the AML
- the IPE Module or Application Module with its appropriate software (including CCR)

In engineering a CCR system, you are really finding the capacity of the system. The capacity of a CCR system depends on the following factors:

- the capacity of the Meridian 1 system
Has the Meridian 1 system enough spare capacity to add CCR? Or, if CCR is already installed, will CCR's portion of the total capacity be large enough to handle your requirements?
- the capacity of the AML
Can the AML handle the messages that will be generated by CCR?
- the capacity of the IPE Module or Application Module
Will the module be able to handle the requirements of CCR?

The capacity of the CCR system is the smallest of these three capacities.

Chapter 2, "Engineering the Meridian 1 for Meridian Link and CCR applications," describes how you can estimate the capacity of the Meridian 1 system.

© 1997 Northern Telecom
All rights reserved

Printed in U.S.A.

Northern Telecom reserves the right to make changes in equipment, design or components as progress in engineering or manufacturing may warrant.

Meridian 1 and SL-1 are trademarks of Northern Telecom. Motorola is a trademark of the Motorola Corporation. MVME products are trademarked by the Motorola Corporation.

This appendix describes how you can estimate the capacity of the CCR system. The accuracy of your estimate depends on how accurately you estimate the number of messages generated by your scripts. This appendix describes how you can optimize a script.

Engineering the capacity of a CCR system

The following factors influence the capacity of a CCR system:

- the Meridian 1 system option (Options 11-81)
- the Meridian 1 system software release
- the portion of Meridian 1 system real time allotted to CCR
- the baud rate of the AML
- the CCR release
- the number of ACD DN's for which statistics are provided by the Meridian 1 system to CCR (see the note below for more information)
- the average number of messages generated per call by active scripts
- the average length of time a call is controlled by CCR (includes the queuing time for QUEUE TO commands and waiting time for WAIT commands)

The following factors *do not* influence the capacity of a CCR system:

- the number of active scripts
- the number of CDNs

The Meridian 1 system provides ACD DN statistics to CCR for all ACD DN's used

- as parameters in intrinsics

For example, the command IDLE AGENTS 8900 causes the Meridian 1 system to send ACD DN statistics on ACD DN 8900

so

that CCR can evaluate the command (true or false)

- as ACD variables in the Variable Table

Calculating the CCR system capacity using the IPE Module or Application Module capacity rating

The IPE Module or Application Module capacity rating is based on the type of processor used in the Single Board Computer (SBC) card.

The assumptions used are listed here:

- the AML baud rate is 19,200 bps (for more information, refer to Chapter 2, “Engineering the Meridian 1 for Meridian Link and CCR applications”)
 - traffic arrives at a steady rate
 - each call generates ten messages on average
- 1 Calculate the portion of the capacity of the Meridian 1 system allotted to CCR. Refer to *Appendix C, “Meridian 1 CPU capacity engineering”, for information on calculating the capacity of a Meridian 1 system.*
 - 2 Compare the CCR portion of the capacity of the Meridian 1 system. If the IPE Module or Application Module contains an SMM167 or MVME167 card, the rating is 16,500 calls per hour.
 - 3 Use the lower value of the IPE Module or Application Module rating (step 2) and the portion of the Meridian 1 capacity (step 1) as the capacity of the CCR system.

Calculating the CCR system capacity using the IPE Module or Application Module performance chart

The IPE Module or Application Module performance chart (shown in Figure 14) is based on the type of processor used in the SBC card. Use this method if your scripts generate more than ten messages per call on average. Refer to the section “Analyzing execution time” and “Analyzing messages per call” later in this chapter for information on how to analyze your scripts to obtain this value. (If you are not sure, use a value of ten messages per call.)

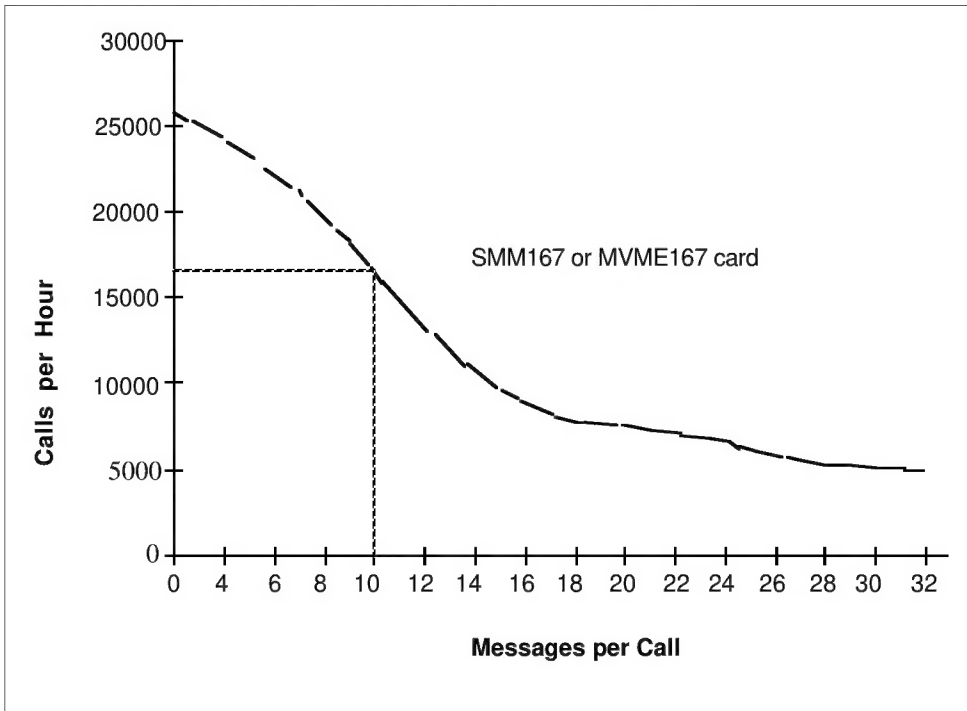
The assumptions used are listed here:

- the AML baud rate is 19,200 bps (for more information, refer to Chapter 2, “Engineering the Meridian 1 for Meridian Link and CCR applications”)
- traffic arrives at a steady rate
- 10 messages per call
- no ACD statistics are used

- 1 Calculate the portion of the capacity (calls per hour) of the Meridian 1 system allotted to CCR.
Refer to Chapter 2, “Engineering the Meridian 1 for Meridian Link and CCR applications,” for information on calculating the capacity of a Meridian 1 system.
- 2 Using the chart shown in Figure 14, read along the horizontal axis to the point that represents the number of messages per call for your system. Draw a vertical line to the curve. From the point where the vertical line intersects with the curve, draw a horizontal line to the vertical axis to obtain a value for the capacity of the IPE Module or Application Module.

The capacity for 10 messages per call is approximately 16,500 calls per hour.
- 3 Use the lower value of the IPE Module or Application Module performance rating (step 2) and the portion of the Meridian 1 capacity (step 1) as the capacity of the CCR system.

Figure 14
CCR performance chart without ACD DN statistics



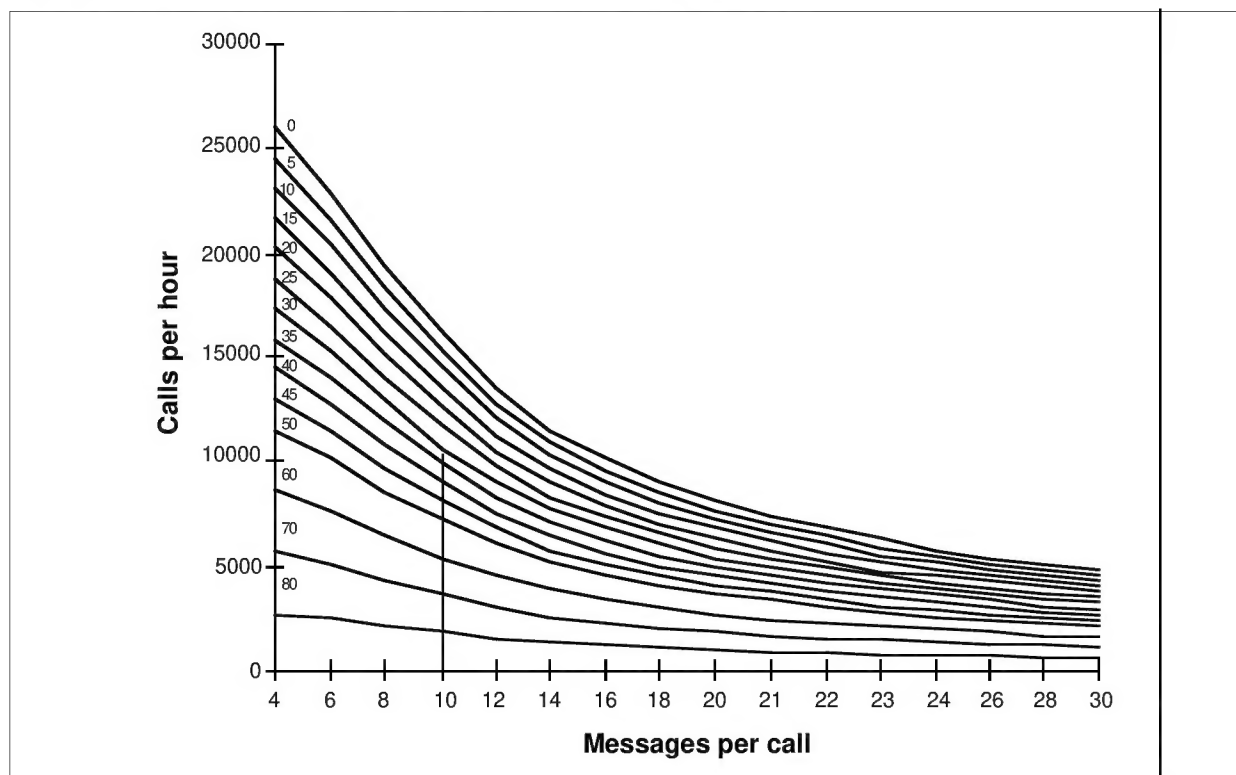
Calculating the CCR system capacity using ACD DN statistics

This method provides a value for the CCR system capacity that is more accurate than the previous two methods but requires an analysis of the scripts to be used and knowledge of the numbered ACD DN station to which statistics are being delivered.

The assumptions used are listed here:

- the AML baud rate is 19,200 bps (for more information, refer to Chapter 2, “Engineering the Meridian 1 for Meridian Link and CCR applications”)
 - traffic arrives at a steady rate
- 1 Calculate the portion of the capacity of the Meridian 1 system allotted to CCR. Refer to Chapter 2, “Engineering the Meridian 1 for Meridian Link and CCR applications,” for information on calculating the capacity of a Meridian 1 system.
 - 2 Calculate the number of ACD DNs for which the Meridian 1 system will provide statistics to CCR. Use the following steps:
 - list all ACD DNs used as a parameter in an intrinsic in any script (for example, ACD DN 8900 in “IDLE AGENTS 8900”)
 - list all ACD DNs in the Variable Table
 - delete any ACD DNs from the second list that appear in the first list
 - count the resulting ACD DNs from the two lists
 - 3 On Figure 15, find the curve that represents the number of ACD DNs closest to the value obtained in step 2. Read along the horizontal axis to find the value that represents the average number of messages per call for your CCR system. Draw a vertical line from the point on the horizontal axis representing the average number of messages per call to the chosen curve. From the point where the vertical line intersects the curve, draw a horizontal line to the vertical axis to obtain the capacity in calls per hour.

Figure 15
CCR performance chart with ACD DN statistics (MVME167)



N The numbers to the right of the vertical axis identify lines with
 0–80 ACD DN's.

t

e

:

For example, assume that, in step 2, you listed 30 ACD DN's. Assume also that your CCR system generates ten messages per call. Figure 15 shows that the capacity is approximately 11,000 calls per hour.

- 4** Use the lower value of the CCR module call capacity (step 3 or 4) and the portion of the Meridian 1 capacity (step 1) as the capacity of the CCR system.

Analyzing scripts

The number of messages generated per call has a major effect upon the capacity of a CCR system. For example, from Figure 15, the capacity of a CCR system with 30 ACD DN's providing statistics and with an average of 10 messages per call is 11,000 calls per hour. The capacity of the same system but with an average of 12 messages per call is 9000 calls per hour. The number of messages per call can vary from as few as three to as many as 35.

The factors that influence how many messages a script will generate are listed:

- the call type — how many paths through the script are there, and what happens on each path?
- the execution time — how long does a call take to complete or how long does the call wait before it is answered or abandoned?
- the length and structure of the script — how many commands are there per call, and how long a delay between commands?

This section describes how you can analyze a script to determine the types of calls it creates and the number of messages that will be generated by that call. This section also describes how you can change the structure to make the script more efficient.

Analyzing call types

The number of paths a call can take through a script identifies the number of *call types* that the script has. Scripts can be simple, with one or two paths that a call can take, or complicated, with several paths that a call can take. Yet, even with very complicated scripts, most calls will use only a few call types, and these call types are usually apparent. You should analyze your scripts to find out how many call types each script has, and what happens to each type. You may want to analyze all call types for simple scripts but only the most-used call types for very complicated scripts. The following examples show the method.

Example 1

```
QUEUE TO 8900
WAIT 20
GIVE RAN ran_route
GIVE MUSIC music_route
```

The script in Example 1 has only a single path. All calls (if they wait for 20 seconds or more) execute all commands. This example has one call type.

Example 2

```
FORCE BUSY IF (Total Queued Calls 8900 > 100)
QUEUE TO 8900
WAIT 20
GIVE RAN ran_route
GIVE MUSIC music_route
```

The script in Example 2 has two paths:

- path 1, in which calls receive a busy signal and drop from CCR control
- path 2, in which calls execute the remaining commands

The script in Example 2 has two call types. If this is a new script in an existing system, you may be able to use historical reports to find out how many calls receive busy signals. If this is a new script in a new system, you must estimate how many calls you expect will receive busy signals.

Example 3

```

GOTO Platinum_Callers IF DNIS = platinum_dnis
GOTO Gold_Callers IF DNIS = gold_dnis
GOTO Regular_Callers
SECTION Platinum_Callers
    QUEUE TO cust_svc WITH PRIORITY 1
    QUEUE TO special_svc WITH PRIORITY 1
    GIVE RAN platinum_ran
    GIVE MUSIC soft_music
    QUIT
SECTION Gold_Callers
    QUEUE TO cust_svc WITH PRIORITY 2
    GIVE RAN gold_ran
    GIVE MUSIC soft_music
SECTION Loop
    WAIT 2
    GOTO Queue_Secondary IF Age Of Call > 30
    GOTO Loop
SECTION Queue_Secondary
    QUEUE TO special_svc WITH PRIORITY 2
    QUIT
SECTION Regular_Callers
    FORCE BUSY IF (Day Of Year = Holiday OR
                  Day Of Week = Weekend) AND
                  (Total Queued Calls cust_svc >
                   (2*Logged Agents cust_svc))
    QUEUE TO cust_svc WITH PRIORITY 3
    GIVE RAN regular_ran
SECTION Regular_Loop
    WAIT 2
    GOTO Change_Priority IF Age Of Call > 60
    GOTO Regular_Loop
SECTION Change_Priority
    QUEUE TO cust_svc WITH PRIORITY 2
    QUIT

```

On first glance at Example 3, you may see three call types; one each for the Platinum, Gold, and Regular Callers. However, if you analyze the script carefully, you will see an additional call type, making a total of four call types.

Call type 1

```

GOTO Platinum_Callers IF DNIS = platinum_dnis
.
.
SECTION Platinum_Callers
    QUEUE TO cust_svc WITH PRIORITY 1
    QUEUE TO special_svc WITH PRIORITY 1
    GIVE RAN platinum_ran
    GIVE MUSIC soft_music
    QUIT

```

Call type 2

```
GOTO Gold_Callers IF DNIS = gold_dnis
.
.
SECTION Gold_Callers
    QUEUE TO cust_svc WITH PRIORITY 2
    GIVE RAN gold_ran
    GIVE MUSIC soft_music
SECTION Loop
    WAIT 2
    GOTO Queue_Secondary IF Age Of Call > 30
    GOTO Loop
SECTION Queue_Secondary
    QUEUE TO special_svc WITH PRIORITY 2
    QUIT
```

If you examine the script, you will see that some regular callers will receive busy signals, and others will not.

Following is the path for the call type for regular callers who do not receive busy signals.

Call type 3

```
GOTO Regular_Callers
.
.
SECTION Regular_Callers
.
.
    QUEUE TO cust_svc WITH PRIORITY 3
    GIVE RAN regular_ran
SECTION Regular_Loop
    WAIT 2
    GOTO Change_Priority IF Age Of Call > 60
    GOTO Regular_Loop
SECTION Change_Priority
    QUEUE TO cust_svc WITH PRIORITY 2
    QUIT
```

Following is the path for the call type for regular callers who receive busy signals.

Call type 4

```
GOTO Regular_Callers
.
.
SECTION Regular_Callers
    FORCE BUSY IF (Day_Of_Year = Holiday OR
                  Day_Of_Week = Weekend) AND
                  (Total_Queued_Calls cust_svc >
                   (2*Logged_Agents cust_svc))
```

Analyzing execution time

After analyzing your script in the previous section to find the number of call types, you should examine each call type to find out how long the call type will take to execute. Also find out how many messages the call type will generate while executing. This section describes how you can do this. This section also describes how you can create a spreadsheet and use the spreadsheet to create charts similar to those shown earlier in this chapter.

CCR commands may be *executed* or *unexecuted*. An executed command can be either an unconditional command (such as QUEUE TO 8900) or a conditional command whose condition evaluates to true (such as QUEUE TO 8900 IF Time of Day = 08:00..17:00 if evaluated during normal working hours).

An unexecuted command is a conditional command whose condition evaluates to false (such as QUEUE TO 8900 IF Time of Day = 08:00..17:00 if evaluated after normal working hours). Each executed CCR command generates a specific number of messages, and many CCR commands have a specific execution time. Table 17 shows the time to execute and the number of messages generated by executed CCR commands.

Table 17
CCR command execution times and messages generated

CCR command	Execution time	Number of messages
GOTO <i>section</i>	0	0
WAIT <i>seconds</i>	number of seconds	0
SECTION <i>label</i>	0	0
QUIT	terminates	0
QUEUE TO <i>acd_dn</i>	0	2
REMOVE FROM <i>acd_dn</i>	0	2
ROUTE TO <i>dn</i>	terminates	2
FORCE BUSY	terminates	2
FORCE DISCONNECT	terminates	2
GIVE RINGBACK	0	2
GIVE SILENCE	0	2
GIVE MUSIC <i>music_route</i>	0	2
GIVE RAN <i>ran_route</i>	length of RAN	3
GIVE IVR <i>ivr_dn</i>	length of IVR session	3

Some commands (such as GOTO and WAIT) do not generate any messages and you can disregard them when you are calculating the message traffic.

In addition to the messages generated per executed command (the command messages), overhead messages may be generated by a call depending on how the call ends. If the call is answered or abandoned, the call generates two overhead messages in addition to the command messages. If a call terminates in a FORCE BUSY, FORCE DISCONNECT, or ROUTE TO command, the call generates one overhead message in addition to the command messages.

The total number of messages generated per call = $m_0 + (m_1 + m_2 + \dots + m_n)$

in which m_0 represents the overhead messages and m_1 through m_n represent the messages for each executed command in the script.

Example 4

```
QUEUE TO 8900
WAIT 20
GIVE RAN ran_route
GIVE MUSIC music_route
```

The script in Example 4 generates 9 messages as follows:

- QUEUE TO 8900 generates 2 messages
- WAIT 20 generates 0 messages
- GIVE RAN ran_route generates 3 messages
- GIVE MUSIC music_route generates 2 messages
- the script generates 2 overhead messages because all calls will be either answered or abandoned

Example 5

```
FORCE BUSY IF (Total Queued Calls 8900 > 100)
QUEUE TO 8900
WAIT 20
GIVE RAN ran_route
GIVE MUSIC music_route
```

The script in Example 5 has two call types as shown by Table 18.

Table 18
Messages generated by Example 5

		Execution Time (secs)	Cumulative Script Execution Time (secs)	Messages per Command	Cumulative Script Messages
Call Type 1	FORCE BUSY	0	0	2	2
Call Type 2	QUEUE TO	0	0	2	2
	WAIT	10	10	0	2
	GIVE RAN	30	40	3	5
	GIVE MUSIC	0	40	2	7

Call type 1 (calls receiving busy signals) generates 3 messages as follows:

- FORCE BUSY IF (Total Queued Calls 8900 > 100) generates 2 messages
- the call type generates one overhead message because it ends in a FORCE BUSY command

Call type 2 (calls queued) generates 9 messages as follows:

- QUEUE TO 8900 generates 2 messages
- WAIT 20 generates 0 messages
- GIVE RAN ran_route generates 3 messages
- GIVE MUSIC music_route generates 2 messages
- the call type generates 2 overhead messages because all calls will be either answered or abandoned

Example 6

```

GOTO Platinum_Callers IF DNIS = platinum_dnis
GOTO Gold_Callers IF DNIS = gold_dnis
GOTO Regular_Callers
SECTION Platinum_Callers
    QUEUE TO cust_svc WITH PRIORITY 1
    QUEUE TO special_svc WITH PRIORITY 1
    GIVE RAN platinum_ran
    GIVE MUSIC soft_music
    QUIT
SECTION Gold_Callers
    QUEUE TO cust_svc WITH PRIORITY 2
    GIVE RAN gold_ran
    GIVE MUSIC soft_music
SECTION Loop
    WAIT 2
    GOTO Queue_Secondary IF Age Of Call > 30
    GOTO Loop
SECTION Queue_Secondary
    QUEUE TO special_svc WITH PRIORITY 2
    QUIT
SECTION Regular_Callers
    FORCE BUSY IF (Day Of Year = Holiday OR
                  Day Of Week = Weekend) AND
                  (Total Queued Calls cust_svc >
                   (2*Logged Agents cust_svc))
    QUEUE TO cust_svc WITH PRIORITY 3
    GIVE RAN regular_ran
SECTION Regular_Loop
    WAIT 2
    GOTO Change_Priority IF Age Of Call > 60
    GOTO Regular_Loop
SECTION Change_Priority
    QUEUE TO cust_svc WITH PRIORITY 2
    QUIT

```

As described in Example 3, the script in Example 6 has four call types. Table 19 shows the messages generated by each call type.

Table 19
Messages generated by Example 6

		Delay per Command (secs)	Cumulative Script Execution Time (secs)	Messages per Command	Cumulative Script Messages
Platinum	QUEUE TO	0	0	2	2
	QUEUE TO	0	0	2	4
	GIVE RAN	20	20	3	7
	GIVE MUSIC	0	20	2	9
Gold	QUEUE TO	0	0	2	2
	GIVE RAN	25	25	3	5

	GIVE MUSIC	0	25	2	7
	WAIT 2	2	27	0	7
	WAIT 2	2	29	0	7
	WAIT 2	2	31	0	7
	QUEUE TO	0	31	2	9
Regular-busy	FORCE BUSY	0	0	2	2
	QUEUE TO	0	0	2	2
Regular-queue	GIVE RAN	15	15	3	5
	WAIT 2	2	17	0	5
	WAIT 2	2	19	0	5
	WAIT 2	2	21	0	5
	WAIT 2	2	23	0	5
	WAIT 2	2	25	0	5
— continued —					

Table 19
Messages generated by Example 6, continued

		Delay per Command (secs)	Cumulative Script Execution Time (secs)	Messages per Command	Cumulative Script Messages
Regular-queue (continued)	WAIT 2	2	27	0	5
	WAIT 2	2	29	0	5
	WAIT 2	2	31	0	5
	WAIT 2	2	33	0	5
	WAIT 2	2	35	0	5
	WAIT 2	2	37	0	5
	WAIT 2	2	39	0	5
	WAIT 2	2	41	0	5
	WAIT 2	2	43	0	5
	WAIT 2	2	45	0	5
	WAIT 2	2	47	0	5
	WAIT 2	2	49	0	5
	WAIT 2	2	51	0	5
	WAIT 2	2	53	0	5

WAIT 2	2	55	0	5
WAIT 2	2	57	0	5
WAIT 2	2	59	0	5
WAIT 2	2	61	0	5
QUEUE TO	0	61	2	7

Table 19 shows that call type 4 (shown in the following series of commands) executes inefficiently. The WAIT 2 statement repeats approximately 23 times in the Regular Loop until the call has aged beyond 60 seconds. During that time, the WAIT 2 statement does no useful work.

```
SECTION Regular_Callers
.
.
QUEUE TO cust_svc WITH PRIORITY 3
GIVE RAN regular_ran
SECTION Regular_Loop
WAIT 2
GOTO Change_Priority IF Age_Of_Call > 60
GOTO Regular_Loop
SECTION Change_Priority
QUEUE TO cust_svc WITH PRIORITY 2
QUIT
```

The following series of commands shows a more efficient way to achieve the same effect:

```
SECTION Regular_Callers
.
.
QUEUE TO cust_svc WITH PRIORITY 3
GIVE RAN regular_ran
WAIT 40
SECTION Regular_Loop
WAIT 2
GOTO Change_Priority IF Age_Of_Call > 60
GOTO Regular_Loop
SECTION Change_Priority
QUEUE TO cust_svc WITH PRIORITY 2
QUIT
```

By adding the WAIT 40 command, the call will have aged to approximately 55 seconds before it enters the Regular Loop and hence the number of executions of the inefficient WAIT 2 command has been reduced dramatically.

Analyzing messages per call

After analyzing your script to find the number of call types, and examining each call type to find out how long the call type will take to execute, you should analyze how many messages the call type will generate while executing. A factor affecting the number of messages generated by a call is the time required by the script to execute. In most cases, commands take very little time to execute and for the purposes of these calculations can be considered to take no time to execute. However, commands such as WAIT, GIVE RAN, and GIVE IVR impose a waiting time on the script. The WAIT command has a waiting time indicated by the command's parameter; the GIVE RAN command has a waiting time equal to the length of the RAN; the GIVE IVR command has a waiting time equal to the length of the IVR session.

Example 7

```
QUEUE TO 8900
WAIT 20
GIVE RAN ran_route
GIVE MUSIC music_route
```

If the RAN is 30 seconds long

- QUEUE TO 8900 generates 2 messages and has 0 execution time
- WAIT 20 generates 0 messages and has 20 seconds execution time
- GIVE RAN ran_route generates 3 messages and has 30 seconds execution time
- GIVE MUSIC music_route generates 2 messages and has 0 execution time

The script in Example 7 can be completed only if the call's execution time is longer than 50 seconds. If a call is answered or abandoned after less than 20 seconds, only the QUEUE TO and WAIT commands execute, generating four messages (two command messages for the QUEUE TO command and two overhead messages because the call is answered or abandoned). If a call is answered or abandoned after 20 to 50 seconds, the QUEUE TO, WAIT, and GIVE RAN commands execute, generating seven messages (five command messages and two overhead messages). If a call waits for more than 50 seconds, the complete script executes, generating nine messages (seven command messages, two overhead messages).

Example 8

```

FORCE BUSY IF (Total Queued Calls > 100)
QUEUE TO 8900
WAIT 20
GIVE RAN ran_route
GIVE MUSIC music_route

```

Table 20 is a spreadsheet showing the execution time and number of messages generated during the execution of the script, assuming the RAN is 30 seconds long.

Table 20
Execution time and messages generated by Example 8

		Execution Time (secs)	Cumulative Script Execution Time (secs)	Messages per Command	Cumulative Script Messages
Call Type 1	FORCE BUSY	0	0	2	2
Call Type 2	QUEUE TO	0	0	2	2
	WAIT	10	10	0	2
	GIVE RAN	30	40	3	5
	GIVE MUSIC	0	40	2	7

Scenario 1

If 20% of calls receive busy signals, and 80% of calls wait for 50 seconds and complete the script, Table 21 is a spreadsheet showing that the average number of messages per call is 7.8.

Table 21

Average execution time for Example 8 (1)

		Number messages/call	% of Call Type
Busy Tone	Script	2	
	Overhead	1	
	Total	3	20%
Queue	Script	7	
	Overhead	2	
	Total	9	80%
		7.8	

You can use this value on the horizontal axis of Figure 15 to find the capacity (number of calls per hour) of the CCR module, assuming that all calls use the script shown in Example 8.

Scenario 2

If, however, 20% of calls receive busy signals, 40% are answered or abandoned in less than 25 seconds, and 40% are complete, Table 22 is a spreadsheet showing that the average number of messages per call is 7.

Like Scenario 1, you can use this value on the horizontal axis of Figure 15.

Table 22
Average execution time for Example 8 (2)

		Length (seconds) and % of Calls				Average msgs/call	% of Call Type
		<25 seconds		Complete			
		#msgs	%	#msgs	%		
Busy Tone	Script	2		2			
	Overhead	1		1			
	Total	3	50%	3	50%	3.0	20%
Queue	Script	5		7			
	Overhead	5		2			
	Total	7	50%	9	50%	8.0	80%
						7.0	

Example 9

```
GOTO Platinum_Callers IF DNIS = platinum_dnis
GOTO Gold_Callers IF DNIS = gold_dnis
GOTO Regular_Callers
SECTION Platinum_Callers
    QUEUE TO cust_svc WITH PRIORITY 1
    QUEUE TO special_svc WITH PRIORITY 1
    GIVE RAN platinum_ran
    GIVE MUSIC soft_music
    QUIT
SECTION Gold_Callers
    QUEUE TO cust_svc WITH PRIORITY 2
    GIVE RAN gold_ran
    GIVE MUSIC soft_music
SECTION Loop
    WAIT 2
    GOTO Queue_Secondary IF Age_Of_Call > 30
    GOTO Loop
SECTION Queue_Secondary
    QUEUE TO special_svc WITH PRIORITY 2
    QUIT
SECTION Regular_Callers
    FORCE BUSY IF (Day_Of_Year = Holiday OR
                  Day_Of_Week = Weekend) AND
                  (Total_Queued_Calls cust_svc >
                   (2*Logged_Agents cust_svc))
    QUEUE TO cust_svc WITH PRIORITY 3
    GIVE RAN regular_ran
SECTION Regular_Loop
    WAIT 2
    GOTO Change_Priority IF Age_Of_Call > 60
    GOTO Regular_Loop
SECTION Change_Priority
    QUEUE TO cust_svc WITH PRIORITY 2
    QUIT
```


Scenario 3

- 20% of all callers are from Platinum Card holders and, of these, 90% are answered within 10 seconds.
- 30% of all callers are from Gold Card holders and, of these, 70% are answered within 30 seconds.
- 5% of all callers are from Regular Card holders and receive busy signals.
- 45% of all callers are from Regular Card holders and, of these, 60% are answered within 60 seconds.

Table 23 is a spreadsheet showing that the average number of messages for this scenario is 7.5 messages per call. Like Scenario 1, you can use this value on the horizontal axis of Figure 15.

Table 23
Average execution time for Example 9

Note: Due to the complexity of this table, we are not able to embed it in this document. To see its contents, you must print the file “520-Table 23.”

Other characteristics that affect traffic calculations

The calculations in this appendix make some assumptions. This section describes characteristics that could affect these assumptions and hence the calculations.

Steady arrival rate CCR traffic

The calculations in this appendix assume that the traffic arrives at a steady rate; that is, the time between the arrival of successive calls is approximately equal. For example, a steady arrival rate of 10,800 calls per hour means three calls arrive at equal intervals per second.

Burst rate CCR traffic

An average hourly call rate of 10,800 calls can also be the result of 800 calls arriving in a burst over the first 30 seconds and the remaining 10,000 calls arriving over the rest of the hour. This type of traffic is called burst rate traffic. The more uneven the rate of arrival, the more the traffic resembles burst rate traffic rather than steady arrival rate traffic.

Effect of burst rate CCR traffic on capacity

All calculations in this appendix assume that traffic arrives at a steady rate. The CCR system can sustain the calculated steady rate traffic indefinitely. The CCR system can also sustain bursts of higher call rates but only for a limited time.

An IPE Module or an Application Module equipped with an MVME167 card can handle a burst of traffic arriving at 10 to 12 calls per second (36,000 to 43,200 calls per hour) for no longer than 30 seconds. The same module can handle a burst of traffic arriving at about 20 calls per second (72,000 calls per hour) for 5 to 6 seconds. A burst of traffic arriving at a rate of 25 calls per second (90,000 calls per hour) or greater overloads the module almost immediately.

Controlling overload

Overload occurs when a burst of traffic has exceeded the capacity of the system. The system handles overload conditions in different ways, depending on the type of overload.

Congestion control

When a call arrives, CCR must respond to the Meridian 1 system within four seconds with a valid treatment for that call. If CCR does not respond in time or with a valid treatment, the call defaults. That is, the call is removed from CCR control, is routed to the ACD DN of the CDN where the call arrived, and is treated as if it entered the ACD DN directly.

If ten successive calls default, the Meridian 1 assumes that the system is congested and prints a CCR001 message followed by a CSA105 message on the maintenance console. From this point, the Meridian 1 system gives all CCR calls (that is, calls arriving at CDNs controlled by CCR) default treatment until the congestion clears. CCR and the Meridian 1 system clear the congestion with a handshake, printing a CCR002 message followed by a CSA104 message on the Meridian 1 maintenance console.

Congestion usually clears within 10 to 20 seconds. No calls are lost, because the Meridian 1 handles them during congestion. After congestion clears, CCR call processing resumes without manual intervention. Infrequent congestion due to an unusually high burst of traffic is considered normal.

CCR flow control

When the AML becomes overloaded (for example, by a burst of traffic), the Meridian 1 system sends overload messages to CCR to control the execution of calls. Depending on the arrival rate of calls, the number of calls waiting, and the configuration of the AML, the Meridian system assigns a flow control level to the overload messages. Table 24 shows the flow control levels, what they mean, and how CCR responds to them.

Table 24
CCR flow control levels

Level	Meaning	CCR action
0	No flow control needed	Continue or resume normal call processing execution.
1	Warning. Inbound traffic was too high during the last time period.	Slow down call execution by waiting between calls.
2	Some messages were lost. Inbound traffic was too high for two successive time periods.	Reject all new calls back to the Meridian 1 system where they will receive default treatment. No new calls will be accepted until level 0 is restored or the link fails.
3	All inbound traffic is lost. If inbound traffic continues, the link will fail. Inbound traffic can resume when level 0 is restored.	New calls will either time out or receive default treatment. CCR cannot communicate with the Meridian 1 system and responds to a level 3 event in a similar way to a link-down event. It cleans up its call database and does nothing until it receives a level 0 message.

CCR activities that do not relate to call execution (such as script installation and removal) are not affected by flow control as long as they do not require communication with the Meridian 1 system.

If the link fails, or if CCR is stopped and restarted, or if CCR receives a Meridian 1 initialization signal, CCR assumes a level 0 condition until the Meridian 1 system sends an overload message with a different level.